Spring Data JPA系列：数据查询（Specification）（二）

写了一系列入门文章之后，博客也有了一些访问量，按照计划，对数据查询进行深入一些的探究，包括

- inner join查询
- 连接对象的属性值查询
- in条件查询
- left join查询
  还是入门级的示例，更深入的用法需要在实际场景中深化。

1、更改Customer类

增加@OneToMany注解的订单对象
需要注意的是，这次增加了Lombok依赖，一个简化对象类定义的插件，详见：
https://projectlombok.org/

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.NamedQuery;
import javax.persistence.*;
import java.util.List;
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@NamedQuery(name="Customer.findByFirstName",query = "select c fro
public class Customer {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String firstName;
    private String lastName;

    //一对多，一个客户对应多个订单，关联的字段是订单里的cId字段
    @OneToMany
    @JoinColumn(name = "cId")
    private List&lt;MyOrder&gt; myOrders;
    public Customer(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    @Override
    public String toString() {
        return String.format(
                "Customer[id=%d, firstName='%s', lastName='%s']",
                id, firstName, lastName);
    }
}
```

2、增加MyOrder类

我的订单对象

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import javax.persistence.*;
import java.math.BigDecimal;
/**
 * Created by Administrator on 2017/7/17 0017.
 */
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class MyOrder {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String code;
    private Long cId;
    private BigDecimal total;
    //实体映射重复列必须设置：insertable = false,updatable = false
    @OneToOne
    @JoinColumn(name = "cId",insertable = false,updatable = false
    private Customer customer;
    @Override
    public String toString() {
        return "MyOrder{" +
                "id=" + id +
                ", code='" + code + '\'' +
                ", cId=" + cId +
                ", total=" + total +
                ", customer=" + customer +
                '}';
    }
}
```

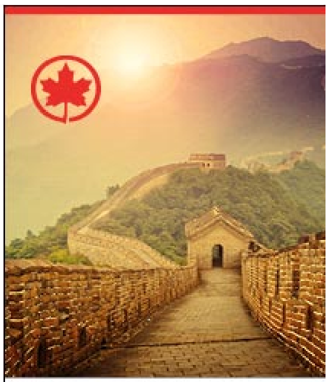
3、新增MyOrderRepository类

这里主要是继承JpaSpecificationExecutor接口，进行Specification查询

```
import com.example.demo.dto.MyOrder;
import org.springframework.data.jpa.repository.JpaSpecificationEx
import org.springframework.data.repository.CrudRepository;
/**
 * Created by Administrator on 2017/7/17 0017.
 */
public interface MyOrderRepository extends JpaSpecificationExecut
}
```

4、新增ShoppingController类

```
import com.example.demo.dto.Customer;
import com.example.demo.dto.MyOrder;
import com.example.demo.repositories.MyOrderRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import javax.persistence.criteria.*;
@Controller
@RequestMapping("/shop")
```



```
public class ShoppingController {
    @Autowired
    private MyOrderRepository myOrderRepository;
    /**
     * 内连接查询
     */
    @RequestMapping("/q1")
    public void specification1(){
        //根据查询结果，声明返回值对象，这里要查询用户的订单列表，所以声明返
        Specification&lt;MyOrder&gt; spec = new Specification&lt;
            //Root&lt;X&gt;  根查询，默认与声明相同
            @Override
            public Predicate toPredicate(Root&lt;MyOrder&gt; root
                //声明并创建MyOrder的CriteriaQuery对象
                CriteriaQuery&lt;MyOrder&gt; q1 = cb.createQuery(
                //连接的时候，要以声明的根查询对象（这里是root，也可以自己
                //Join&lt;Z,X&gt;是Join生成的对象，这里的Z是被连接的对
                //  连接的属性字段是被连接的对象在目标对象的属性，这里是我
                //join的第二个参数是可选的，默认是JoinType.INNER(内连接
                Join&lt;Customer,MyOrder&gt; myOrderJoin = root.j
                //用CriteriaQuery对象拼接查询条件，这里只增加了一个查询
                q1.select(myOrderJoin).where(cb.equal(root.get("c
                //通过getRestriction获得Predicate对象
                Predicate p1= q1.getRestriction();
                //返回对象
                return p1;
            }
        };
        resultPrint(spec);
    }
    /**
     * 增加查询条件，关联的对象Customer的对象值
     */
    @RequestMapping("/q2")
    public void specification2(){
        Specification&lt;MyOrder&gt; spec = new Specification&lt;
            @Override
            public Predicate toPredicate(Root&lt;MyOrder&gt; root
                CriteriaQuery&lt;MyOrder&gt; q1 = cb.createQuery(
                Join&lt;Customer,MyOrder&gt; myOrderJoin = root.j
                q1.select(myOrderJoin)
                        .where(
                                cb.equal(root.get("cId"),1),//cId
                                cb.equal(root.get("customer").get
                        );
                Predicate p1= q1.getRestriction();
                return p1;
            }
        };
        resultPrint(spec);
    }
    /**
     * in的条件查询
     * 需要将对应的结果集以root.get("attributeName").in(Object.. valu
     * values支持多个参数，支持对象（Object），表达式Expression&lt;?&gt
     */
    @RequestMapping("/q3")
    public void specification3(){
        Specification&lt;MyOrder&gt; spec = new Specification&lt;
            @Override
            public Predicate toPredicate(Root&lt;MyOrder&gt; root
                CriteriaQuery&lt;MyOrder&gt; q1 = cb.createQuery(
                Join&lt;Customer,MyOrder&gt; myOrderJoin = root.j
                q1.select(myOrderJoin)
                        .where(
                                cb.equal(root.get("cId"),1)
                                ,root.get("id").in(1,2,4)
                        );

                Predicate p1= q1.getRestriction();
                return p1;
            }
        };
        resultPrint(spec);
    }
    /**
     * 左外链接查询，对比inner join，
     * 这里只是改了一个参数，将JoinType.INNER改成JoinType.LEFT
     *
     * 注意，当前示例不支持JoinType.RIGHT，用的比较少，没有探究
     */
    @RequestMapping("/q4")
    public void specification4(){
        Specification&lt;MyOrder&gt; spec = new Specification&lt;
            @Override
            public Predicate toPredicate(Root&lt;MyOrder&gt; root
                CriteriaQuery&lt;MyOrder&gt; q1 = cb.createQuery(
                Join&lt;Customer,MyOrder&gt; myOrderJoin = root.j
                q1.select(myOrderJoin).where(cb.equal(root.get("c
                Predicate p1= q1.getRestriction();
                return p1;
            }
        };
        resultPrint(spec);
    }
    /***
    *输出分页信息
    **/
    private void resultPrint(Specification&lt;MyOrder&gt; spec) {
        //分页查询
        Pageable pageable = new PageRequest(0,10, Sort.Direction.
        //查询的分页结果
        Page&lt;MyOrder&gt; page =myOrderRepository.findAll(spec,
        System.out.println(page);
        System.out.println(page.getTotalElements());
        System.out.println(page.getTotalPages());
        for (MyOrder c:page.getContent()){
            System.out.println(c.toString());
        }
    }

}
```

内容已经写进注释了，请读源码，有问题请留言。

5、测试

1）、内连接查询及结果：

- URL:http://localhost:8080/shop/q1
- 结果：

```
Hibernate: select myorder0_.id as id1_1_, myorder0_.c_id as c_id
Hibernate: select customer0_.id as id1_0_0_, customer0_.first_na
```

```
Page 1 of 1 containing com.example.demo.dto.MyOrder instances
5
1
MyOrder{id=5, code='123455', cId=1, total=55.23, customer=Custom
MyOrder{id=4, code='123459', cId=1, total=9.99, customer=Custome
MyOrder{id=3, code='123458', cId=1, total=11.90, customer=Custom
MyOrder{id=2, code='123457', cId=1, total=20.90, customer=Custom
MyOrder{id=1, code='123456', cId=1, total=11.10, customer=Custom
```

2）、关联对象条件查询及结果：

- URL:http://localhost:8080/shop/q2
- 结果：

```
Hibernate: select myorder0_.id as id1_1_, myorder0_.c_id as c_id
Hibernate: select customer0_.id as id1_0_0_, customer0_.first_na
Page 1 of 1 containing com.example.demo.dto.MyOrder instances
5
1
MyOrder{id=5, code='123455', cId=1, total=55.23, customer=Custom
MyOrder{id=4, code='123459', cId=1, total=9.99, customer=Custome
MyOrder{id=3, code='123458', cId=1, total=11.90, customer=Custom
MyOrder{id=2, code='123457', cId=1, total=20.90, customer=Custom
MyOrder{id=1, code='123456', cId=1, total=11.10, customer=Custom
```

3）、in条件查询测试及结果：

- URL:http://localhost:8080/shop/q3
- 结果：

```
Hibernate: select myorder0_.id as id1_1_, myorder0_.c_id as c_id
Hibernate: select customer0_.id as id1_0_0_, customer0_.first_na
Page 1 of 1 containing com.example.demo.dto.MyOrder instances
3
1
MyOrder{id=4, code='123459', cId=1, total=9.99, customer=Custome
MyOrder{id=2, code='123457', cId=1, total=20.90, customer=Custom
MyOrder{id=1, code='123456', cId=1, total=11.10, customer=Custom
```

4）、左外连接测试及结果：

- URL:http://localhost:8080/shop/q4
- 结果：

```
Hibernate: select myorder0_.id as id1_1_, myorder0_.c_id as c_id
Hibernate: select customer0_.id as id1_0_0_, customer0_.first_na
Page 1 of 1 containing com.example.demo.dto.MyOrder instances
5
1
MyOrder{id=5, code='123455', cId=1, total=55.23, customer=Custom
MyOrder{id=4, code='123459', cId=1, total=9.99, customer=Custome
MyOrder{id=3, code='123458', cId=1, total=11.90, customer=Custom
MyOrder{id=2, code='123457', cId=1, total=20.90, customer=Custom
MyOrder{id=1, code='123456', cId=1, total=11.10, customer=Custom
```

参考：
官方文档:https://docs.spring.io/spring-data/jpa/docs/current/reference/html
API官方文档:http://docs.spring.io/spring-data/data-jpa/docs/current/api/
JPQL文档:http://www.blogjava.net/calmJava/archive/2011/04/01/347450.html

DEMO示例：https://github.com/icnws/spring-data-jpa-demo

标签： HTTPS 测试 HTTP Spring Data